Amendments to the Claims:

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously Presented) A memory access method for use in a memory management unit (MMU) of a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system and MMU, the method comprising:

A) initiating a memory access instruction concerning a selected virtual address having an associated context identifier;

B) searching a translation lookaside buffer (TLB) for a TLB entry having the selected virtual address;

C) wherein if a TLB entry having the selected virtual address is found in the TLB:

accessing an associated physical address translation from the TLB entry and executing the memory access instruction;

D) wherein if the TLB entry having the selected virtual address is NOT found in the TLB:

i) determining whether a translation table entry (TTE) corresponding with the selected virtual address is available in secondary memory assets of the system to have memory access instructions performed thereon, wherein said determining (D i) includes testing the associated context identifier to determine the availability of the TTE to have a memory access instruction performed thereon,

E) wherein if the TTE is available:

i) accessing the TTE from the secondary memory assets;

ii) updating at least one of the TLB and the secondary memory assets with information from the TTE; and

iii) returning to A) initiating a memory access instruction; and

F) wherein if the TTE is not available:

i) selectively pausing the method until the TTE becomes available; and

ii) returning to A) initiating a memory access instruction.

2. (Canceled)

3. (Previously Presented) The memory access method as in Claim 1 wherein testing the associated context identifier is performed prior to accessing the TTE from the secondary memory assets.

4. (Original) The memory access method as in Claim 1 wherein updating at least one of the TLB and the secondary memory assets with information from the TTE includes updating with information including physical address information.

5. (Original) The memory access method as in Claim 1 wherein updating at least one of the TLB and the secondary memory assets with information from the TTE includes updating with information including attribute information.

6. (Original) The memory access method as in Claim 1 wherein when the selected virtual address is NOT found in the TLB, said D)(i) determining whether the TTE in secondary memory assets is available comprises:

invoking a TLB miss handler to obtain a TTE that corresponds to the selected virtual address having the associated context identifier, the miss handler conducting the operations of:

a) testing the context identifier, prior to accessing secondary memory assets, to determine if the TTE is available to have memory access instructions performed thereon,

wherein said testing determines that the TTE is available,

i) accessing the TTE from the secondary memory assets;

ii) updating the at least one of TLB and the secondary memory assets with information from the TTE as needed;

iii) returning to A) initiating a memory access instruction;

wherein said testing determines that the TTE is not available,

iv) invoking a miss exception handler to facilitate resolution of a miss exception event that resulted in said unavailability; and

v) returning to A) initiating a memory access instruction.

7. (Previously Presented) The method of Claim 6, wherein a)(iv) invoking a miss exception handler to resolve a miss exception event that resulted in said unavailability includes:

determining the nature of the miss exception event;

selectively pausing the operation of the miss exception handler until the miss exception event has been resolved;

facilitating the resolution of the miss exception event; and

returning to A) initiating a memory access instruction.

8. (Previously Presented) A computer readable media including computer program code for operating a memory management unit (MMU) of a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system and MMU, the computer readable media including:

A) computer program code instructions for initiating a memory access instruction concerning a selected virtual address having an associated context identifier;

B) computer program code instructions for searching a translation lookaside buffer (TLB) for a TLB entry having the selected virtual address;

C) wherein the computer program code instructions find the TLB entry having the selected virtual address in the TLB, the code implements instructions for:

accessing an associated physical address translation from the TLB entry;

executing the memory access instruction using the associated physical address to complete a process;

D) wherein the computer program code instructions do NOT find the TLB entry having the selected virtual address in the TLB, the code implements instructions for invoking a TLB miss handler to obtain a translation table entry (TTE) that corresponds to the selected virtual address and associated context identifier, the miss handler executing computer program code instructions for:

i) determining whether a TTE in secondary memory assets having the selected virtual address and associated context identifier is available to have memory access instructions performed thereon, wherein said determining (D i) includes testing the associated context identifier to determine the availability of the TTE to have a memory access instruction performed thereon,

E) wherein if the computer program code instructions determine that the TTE is available, the code implements:

i) computer program code instructions for accessing the TTE from the secondary memory assets;

ii) computer program code instructions for updating the at least one of the TLB and the secondary memory assets with information from the TTE; and

iii) computer program code instructions for returning to A) initiating a memory access instruction; and

F) wherein if the computer program code instructions determine that the TTE is not available, the code implements:

i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event that resulted in said unavailability; and

ii) computer program code instructions for returning to A) initiating a memory access instruction.

9. (Currently Amended) The computer readable media including computer program code of Claim 8, wherein F)(i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event includes:

computer program code instructions for determining the nature of the miss exception event;

computer program code instructions for selectively pausing the operation of the miss exception handler ~~as necessary~~ until the miss exception event has been resolved; and

computer program code instructions for resolving the miss exception event.

10. (Previously Presented) The computer readable media including computer program code of Claim 8, wherein F)(i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event includes computer program code instructions for invoking a miss exception handler without issuing cross calls that effectively halt one or more CPU's in the multi-processor computing system.

11. (Previously Presented) A computer system comprising:

a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system;

each CPU having a memory cache configured to include a translation lookaside buffer (TLB) having entries configured to include virtual addresses and associated context identifiers for those virtual addresses;

secondary memory assets that include a translation storage buffer and page tables, said secondary memory assets resources including at least one translation table entry (TTE);

a memory management unit (MMU) that includes:

a TLB miss handler that enables:

searching the secondary memory assets to find a selected TTE that is related to a selected virtual address and associated context identifier when the selected virtual address and associated context identifier cannot be located in the TLB, and

testing the associated context identifier to determine if the TTE is available to have a memory access instruction executed thereon, wherein said testing occurs before the searching of the secondary memory assets; and

a miss exception handler that enables:

determining the nature of a TLB miss exception that renders the TTE unavailable;

determining that the miss exception has been resolved; and

selectively pausing the operation of the miss exception handler, if needed, until the miss exception is resolved.

12. (Previously Presented) The computer system of Claim 11 wherein the MMU is configured so that TLB misses and TLB miss exceptions are resolved without issuing cross calls that effectively halt one or more CPU's in the multi-processor computing system.

13. (Previously Presented) A method of handling translation lookaside buffer (TLB) miss exceptions in a memory management unit of a multi-processor computer system having a plurality of Central Processing Units (CPU's), the method comprising:

initiating an access instruction;

determining that a TLB miss exception event has occurred, wherein determining that a TLB miss exception event has occurred includes testing a context identifier for the affected virtual address to determine whether a TLB miss exception has occurred;

invoking a miss exception handler;

determining the nature of the TLB miss exception event;

resolving the miss exception event; and

returning to initiating an access instruction.

14. (Previously Presented) The method of Claim 13 wherein the resolving of the miss exception event includes pausing the miss exception handler until the miss exception event is resolved.

15. (Canceled)

16. (Currently Amended) The method of Claim [[15]] 13 wherein determining the nature of the TLB miss exception event includes testing the context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource; ~~and~~

wherein determining the nature of the TLB miss exception event includes testing a context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource;

wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event; and

wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event.

17. (Canceled)

18. (Currently Amended) The method of Claim [[17]] 16 wherein said testing the context identifier determines that said unavailability results from a miss exception due to an unassigned context identifier, and

wherein resolving the unassigned context identifier miss exception event further includes the steps of:

assigning a context identifier value to a virtual address space that is to be associated with a process that contains the virtual address;

assigning selected portion of memory to a TSB that is to be associated with the virtual address space having the assigned context identifier;

updating at least one of secondary memory assets and the TLB with TSB and context identifier information; and

returning to initiating an access instruction.

19. (Currently Amended) The method of Claim [[17]] 16 wherein said testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the affected virtual address maps to a shared memory resource, and

wherein resolving the miss exception event further includes the steps of:

determining if the shared memory resource is locked;

if the shared memory resource is not locked,

returning to initiating an access instruction; and

if the shared memory resource is locked,

pausing the exception handler until the shared memory resource becomes unlocked; and

returning to initiating an access instruction when the shared memory resource becomes unlocked.

20. (Original) The method of Claim 19 wherein said shared memory resource comprises a translation table entry (TTE).

21. (Currently Amended) The method of Claim [[17]] 16 wherein testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the virtual address is associated with a TSB that is undergoing a resizing operation; and

wherein resolving the miss exception event further includes the steps of:

A) determining if the virtual address space of the TSB undergoing resizing is locked;

1) in a case wherein the virtual address space of the TSB undergoing resizing is locked,

i) pausing the exception handler until the virtual address space of the TSB undergoing resizing becomes unlocked;

ii) locking the virtual address space of the TSB undergoing resizing;

2) in a case wherein the virtual address space of the TSB undergoing resizing is unlocked,

i) locking the virtual address space of the TSB undergoing resizing;

B) once the virtual address space of the TSB undergoing resizing has been locked by one of 1)(ii) and 2(i), determining whether the TSB undergoing resizing has been assigned a specified location in memory;

1) if the TSB undergoing resizing has been assigned a specified location in memory,

i) releasing the lock on the virtual address space of the TSB undergoing resizing; and

ii) returning to initiating an access instruction;

2) if the TSB undergoing resizing has not been assigned a specified location in memory,

i) assigning a specified portion of memory to the TSB undergoing resizing;

ii) releasing the lock on the virtual address space of the TSB undergoing resizing; and

iii) returning to initiating an access instruction.

22. (Previously Presented) A method of accessing translation table entries (TTE's) in secondary memory assets of a memory management unit of a multi-processor computer system, the method comprising:

A) requesting that a translation be found for a selected virtual address having an associated context identifier wherein the translation is to be found in a secondary memory asset of a memory management unit;

B) testing the associated context identifier, prior to searching a secondary memory asset, to determine whether a TTE corresponding to the virtual address and the associated context identifier is available to have a memory access instruction executed upon it;

1) wherein if testing determines the TTE is available to have a memory access instruction executed upon it:

locating the TTE using the virtual address and context identifier;

accessing the TTE from the secondary memory asset;

updating a translation lookaside buffer (TLB) and the secondary memory asset as needed;

returning to initiating an access instruction;

2) wherein if testing determines the TTE is not available to have a memory access instruction executed upon it:

determining a source of unavailability;

resolving the unavailability; and

returning to initiating an access instruction.

23. (Original) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table,

wherein locating the TTE using the virtual address and context identifier include first searching the TSB for a TTE corresponding with the virtual address and associated context identifier and then if a TTE corresponding with the virtual address and associated context identifier is not found in the TSB, locating a TTE corresponding with the virtual address and associated context identifier in a page table;

and

wherein updating, when the TTE is located in a page table comprises updating the TLB to include the TTE and updating the TSB to include the TTE.

24. (Previously Presented) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table, and

wherein resolving the unavailability includes:

determining the nature of the unavailability; and

where it is determined that the cause of the unavailability is a TSB resizing operation that affects the virtual address for which the translation is sought;

pausing until the TSB resizing operation is completed; and

returning to initiating an access instruction.

25. (Previously Presented) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table, and

wherein resolving the unavailability includes:

determining the nature of the unavailability; and

where it is determined that the cause of the unavailability is a demapping operation for a TTE that is shared by the virtual address for which the translation is sought,

pausing until the TTE demapping operation is completed; and

returning to initiating an access instruction.

26. (Currently Amended) A method of accomplishing memory management of miss exceptions in a memory management unit of a multi-processor computer system, the method comprising;

determining that a miss exception event has occurred, wherein the miss exception event concerns one of: an unassigned context identifier event, a memory access event changing a shared memory resource, and a ~~TSB~~ translation storage buffer (TSB) resizing event; [[and]]

resolving the miss exception event in accordance with a miss event resolution protocol suitable for resolving the received miss exception event; and

wherein said determining determines that the miss exception event comprises an instruction to change a translation table entry (TTE) that is shared by more than one

virtual address space wherein each virtual address space has an associated context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

identifying virtual address spaces that share the same TTE;

activating a lock for each virtual address space that shares the same TTE to prevent other processes from accessing the TTE while the lock is activated;

changing the corresponding context identifier for each virtual address space that shares the same TTE thereby making the associated TTE unavailable to have memory access instructions performed thereon;

performing the changes on the TTE;

releasing the locks on each locked virtual address space; and

freeing the corresponding context identifiers for each affected virtual address space.

27. (Canceled)

28. (Original) The method of Claim 26 wherein said determining determines that the miss exception event comprises a miss exception event comprises an instruction to resize a translation storage buffer (TSB); and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

activating a lock for the TSB to prevent other processes from accessing entries in the TSB while the lock is activated;

changing the corresponding context identifier to indicate that the TSB virtual address space is unavailable to have memory access instructions performed thereon;

resizing the TSB;

changing the corresponding context identifier back to its original configuration; and

releasing the lock on the TSB.

29. (Original) The method of Claim 26 wherein said determining determines that the miss exception event comprises an instruction concerning an unassigned context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

assigning a context identifier for a virtual address space associated with a TSB; and

assigning a portion of memory to the TSB.